Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

2018-03

Collecting cyberattack data for industrial control systems using honeypots

Hyun, Dahae

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/58316

Copyright is reserved by the copyright owner.

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

COLLECTING CYBERATTACK DATA FOR INDUSTRIAL CONTROL SYSTEMS USING HONEYPOTS

by

Dahae Hyun

March 2018

Thesis Advisor: Neil C. Rowe
Co-Advisor: Thuy D. Nguyen

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE COLLECTING CYBERATTACK DATA FOR INDUSTRIAL CONTROL SYSTEMS USING HONEYPOTS		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Dahae Hyun			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

Operational technology, information technology for industrial control systems, has advanced more slowly in security than other kinds of information technology. To aid the discovery of indicators of compromise for industrial control systems, this thesis tested a specialized honeypot, Conpot. Conpot is an open-source low-interaction honeypot that simulates an industrial control system such as a power plant and collects information on cyberattacks. We created parsers to extract its log data for use as indicators of compromise. Conpot provided information such as Internet Protocol (IP) addresses, transmission control protocol or user datagram protocol (TCP/UDP) ports, and basic protocol-specific data. While this was useful for identifying the protocols most frequently attacked and the countries of origin of attacks, we recommend using a high-interaction honeypot to generate more effective indicators of compromise.

14. SUBJECT TERMS honeypots, industrial control systems, indicators of compromise, Conpot			**15. NUMBER OF PAGES** 67
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

COLLECTING CYBERATTACK DATA FOR INDUSTRIAL CONTROL SYSTEMS USING HONEYPOTS

Dahae Hyun
Civilian, National Science Foundation Scholarship for Service Recipient
B.A., University of California, San Diego, 2014

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2018

Approved by: Neil C. Rowe
Thesis Advisor

Thuy D. Nguyen
Co-Advisor

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Operational technology, information technology for industrial control systems, has advanced more slowly in security than other kinds of information technology. To aid the discovery of indicators of compromise for industrial control systems, this thesis tested a specialized honeypot, Conpot. Conpot is an open-source low-interaction honeypot that simulates an industrial control system such as a power plant and collects information on cyberattacks. We created parsers to extract its log data for use as indicators of compromise. Conpot provided information such as Internet Protocol (IP) addresses, transmission control protocol or user datagram protocol (TCP/UDP) ports, and basic protocol-specific data. While this was useful for identifying the protocols most frequently attacked and the countries of origin of attacks, we recommend using a high-interaction honeypot to generate more effective indicators of compromise.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

BACnet	building automation and control network
CIP	common industrial protocol
CSV	comma separated value
DNP3	distributed network protocol
EtherNet/IP	Ethernet industrial protocol
GB	gigabyte
HTTP	hypertext transfer protocol
ICCP	inter-control center communications protocol
ICS-CERT	Industrial Control Systems Cyber Emergency Response Team
ID	identification
IEC-104	International Electrotechnical Commission 60870 Standards protocol
IP	Internet protocol
IPMI	intelligent platform management interface
MB	megabyte
MBAP	MODBUS on TCP/IP application data unit
NPS	Naval Postgraduate School
OSSEC	open source host-based intrusion detection system
PLC	programmable logic controller
RAM	random access memory
S7comm	S7 communication
SNMP	simple network management protocol
TCP	transmission control protocol
TCP/IP	transmission control protocol/ Internet protocol
TFTP	trivial file transfer protocol
UDP	user datagram protocol
URL	uniform resource locator
US-CERT	United States Computer Emergency Readiness Team
VM	Virtual Machine
XMPP	extensible messaging and presence protocol

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Dr. Rowe and Professor Nguyen for their assistance, guidance, and patience.

I would like to thank all my friends at the Naval Postgraduate School for the laughter and tears that made the two years fly by. Cervando, thank you for joining me in all the de-stress eating sessions and being my "Vanilla Ice" to solve my initial thesis problems. Liz, thank you for the sisterly support you have given me and making me more social. Devon and Jay, you guys are simply the best. Let's meet up in Colorado Springs one day. Johanna, Jonathan, Jennifer, and Casi. I don't think I would've survived NPS without you guys.

Casey, thank you for sending me your love and support all the way from Massachusetts. Thank you for always reminding me that we are engineers and we can figure out anything.

David, thank you for encouraging me to become tougher every day. You made (and continue to make) everything amazing. Because of you, what should have been the toughest months of graduate school became the best months of my life. Thank you for all the adventures. Thank you for taking care of me when I was sick. Thank you for bringing me Pho, ice cream, and Chex Mix when I was feeling down. But still, what would you do without me?

Most importantly, I would like to thank my family for their support and love. I would not have been able to get this far without them. 오빠, thank you for always being there to take care of me. I am proud to be your little sister. 아빠, thank you for all the support and encouragement you have given me. I will forever be your 오구구. 세상에서 제일 사랑하는 우리 엄마. 당당하고 멋진 딸이 될수 있도록 날 키우느라 고생많았던 엄마… 너무 너무 사랑하고 감사합니다.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

According to Knapp and Langill [1], industrial control systems are the main control systems to monitor and manipulate the operational technology used for critical infrastructures such as electric-power grids, nuclear reactors, and public-transportation systems. While information technology has experienced a rapid growth, operational technology has not advanced as much. Even up to present day, industrial control systems often use legacy devices and using legacy protocols, many proprietary, without much security.

Initially, industrial control systems operated by communicating with only their assigned devices without a need to communicate with any other systems outside of their "bubble." Being contained in the bubble gave a false sense of security and physical security was considered sufficient. Many vulnerabilities inevitably became present as more industrial control systems became TCP/IP-enabled and connected to the Internet, while security measures were not sufficiently improved.

A. MOTIVATION

A tool to improve information-technology security is to apply "indicators of compromise" to detect possible threats. Currently, their use is uncommon for industrial control systems and sharing of open-source indicators is limited. Some vendors recognize that indicators can be a means of intrusion detection but do not actively market products as they do for traditional systems because the demand for them is small. Today, security professionals share known indicators of compromise for industrial control systems through the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) by analyzing malware samples that have been found in industrial control systems. This method is limited in that it is performed after compromise. By the time such analysis is shared, attackers are well on their way to finding new ways to attack.

Knapp and Langill [1] report that industrial control systems are tempting targets because an attack on a critical infrastructure such as the power grid can have a catastrophic effect on the livelihood of individuals and the operation of business and

government. Despite the large potential loss, industrial control systems remain vulnerable due to a common misconception outlined by Knapp and Langill that security of industrial control systems should be treated just like other information security. They also add that applying the same information-technology security measures for industrial control systems is difficult because they often rely on proprietary software and protocols, legacy devices, and outdated operating systems. They state further that much vendor-provided software used by industrial control systems may not be as up-to-date as its information-technology counterpart, and many of their network protocols are known to be insecure. Systems face more difficulties with patching and updating regularly since they require in-depth testing and scheduled maintenance times to apply these updates or patches [2]. With time, vendors may no longer provide support for their software or switch to different protocols and it would be very costly to replace the obsolete devices. In addition, such software may depend on outdated operating systems that still communicate in plain text rather than with secure-shell services.

B. OBJECTIVES

Taking these premises into consideration, our goal for this research is to explore signatures that may be used as indicators of compromise on a simulated industrial control system by gathering information that can be used to prevent future attacks using honeypots. Honeypots are machines and devices that simulate the attackers' desired targets to collect information on their activities. By testing a honeypot specifically for industrial control systems, we can gather data on methods of attack as a whole, how different protocols have different attack vectors, and the current trends in malicious activity.

Our research used Conpot [3] to test the effectiveness of extraction of signatures for indicators of compromise using honeypots. Conpot is a low-interaction honeypot that simulates an industrial control system using a set of templates. We used the default template of Conpot to simulate an electric-power plant and collect attackers' activities on several protocols. After the information was collected, we used a number of protocol-

specific parsers to filter the attacker activity logs to obtain useful indicators of compromise.

C. THESIS ORGANIZATION

Our thesis is organized as follows:

- Chapter II discusses the background for honeypots, Conpot, seven industrial-control-system protocols that were used for our honeypot, indicators of compromise, and previous studies.
- Chapter III describes the problem we attempted to solve.
- Chapter IV outlines our experiment design and methodology.
- Chapter V analyzes collected data and its significance.
- Chapter VI summarizes our conclusion and introduces possible future studies.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

A. INDUSTRIAL CONTROL SYSTEMS

Knapp and Langill [1] explain that industrial control systems monitor and control physical processes in critical infrastructures such as electric-power grids, nuclear reactors, and public transportation systems. They further write that industrial control systems were isolated from the business network and physical security was the only security protection. With the growth in technology and business needs, especially in the 1990s, operational technology became more integrated with information technology. To share data produced by industrial control systems with the business network, removing the air gap that once protected industrial control systems from the rest of the Internet has become the norm since the focus now is data availability rather than confidentiality or integrity.

In addition to these weaknesses, industrial control systems differ from traditional information technology in that many of its devices use vendor-specific protocols to communicate. For example, information technology devices across all vendors use the standardized HTTP protocol to request for or respond with resources. For industrial control systems, depending on the vendor of the device being used, different proprietary protocols may be used for the same communication functions. Knapp and Langill conclude that the lack of uniformity and transparency complicates the security of industrial control systems.

B. HONEYPOTS

To shift the focus in information security from being reactive to proactive, honeypots are helpful. A honeypot is hardware and/or software that simulates an attractive target for attackers as bait to obtain intelligence from attackers [4]. Honeypots offer low false-positive rates and minimal resources by only gathering information when an interaction is taking place, which by definition is unauthorized. There are three major types of honeypots: low-interaction, medium-interaction, and high-interaction [5].

Low-interaction honeypots are the simplest form of honeypots and they typically respond to simple protocols such as HTTP without any doing actual commands or services. They limit the interaction with the attackers and usually simulate the first steps of network protocols without allowing the attackers to log in to the actual machines, making them safe to use. This thesis will test the low-interaction honeypot tool Conpot. Medium-interaction honeypots provide more replies besides the simple handshake of protocols, but still, do not provide operating systems services. High-interaction honeypots are useful for research because they provide an actual operating system and services that allows detailed interaction with the attackers [6]. One example of such an industrial control systems honeypot is GridPot [7], a "symbolic cyber-physical honeynet framework" that is built on top of Conpot to emulate the protocols used for electric grids [8]. High-interaction honeypots are more complicated to install and maintain than low-interaction honeypots [9]. High-interaction honeypots also require a higher level of monitoring since a security compromise may allow control of the operating systems by attackers.

If multiple honeypots are used to create a network, a honeynet is formed [10]. Honeynets are more effective in deceiving the attackers since the complexity and diversity grow with each additional honeypot.

C. CONPOT

Conpot is an open-source low-interaction honeypot that was developed in 2013 [3]. Conpot acts as a master server for commonly used industrial-control-systems network protocols and provides multiple templates that simulate simple forms of them [11]. Conpot offers four different templates:

- "Default template," which simulates an electric-power plant using Siemens SIMATIC S7-200 Programmable Logic Controllers that communicate with at least two slaves [12].
- "Guardian_ast," which simulates the Guardian AST tank-monitoring system typically used in gas stations [13].

- "IPMI," which simulates a basic system using the Intelligent Platform Management Interface (IPMI).
- "Kamstrup_382," which simulates a system for the Kamstrup 382 electricity meter.

In this research, the default template of Conpot was used with seven protocols described as follows.

1. HTTP

The Hypertext Transfer Protocol (HTTP) is an application-layer protocol that allows network HTTP servers (especially Web servers) to respond to the requests made by clients. Servers use default port number 80 to listen for incoming connections [14]. Clients must first establish a connection with the server using the Transmission Control Protocol (TCP) before sending a request. Once the connections are established, clients can request pages using HTTP request messages and the server will provide them using HTTP response messages.

HTTP request messages can be GET, POST, HEAD, PUT, and DELETE. The GET method retrieves pages from the resources, the POST method sends pages, the HEAD method requests HTTP headers, the PUT method uploads pages, and the DELETE method gets rid of a particular page. HTTP responses provide status codes and header information as well as requested pages. HTTP-response status codes in 200s range indicate success, in the 300s range indicate redirection, and in the 400s range indicate errors for status codes.

2. MODBUS over TCP/IP

MODBUS is an application-layer open-source protocol that has been used for industrial control systems since 1979 [15]. MODBUS is a serial protocol used between "master" and "slave" devices to transmit information. The master device must start the transaction with function codes and inform the slaves of which actions to take. These actions are defined by function codes ranging from 1 to 255, with codes from 128 to 255

reserved for exception responses. In a standard MODBUS network, one master can communicate with up to 247 slave devices with addressing of 1 to 247.

Eventually, MODBUS could communicate over TCP/IP using port number 502. Conpot uses MODBUS over TCP/IP, and it handles four types of messages between devices connected over EtherNet TCP/IP networks: request, response, indication, and confirmation [16]. The MODBUS manual states that client must first send a request to start a transaction and an appropriate response will be sent from the master or server device. An indication message is sent by the client to the server and a confirmation message is sent by the server to the client. In MODBUS over TCP/IP, traditional request and response data is encapsulated with a MODBUS TCP/IP header known as the MODBUS Application Data Unit shown in Figure 1. It contains information such as transaction identifier, protocol identifier, length, and unit identifier (Figure 2).

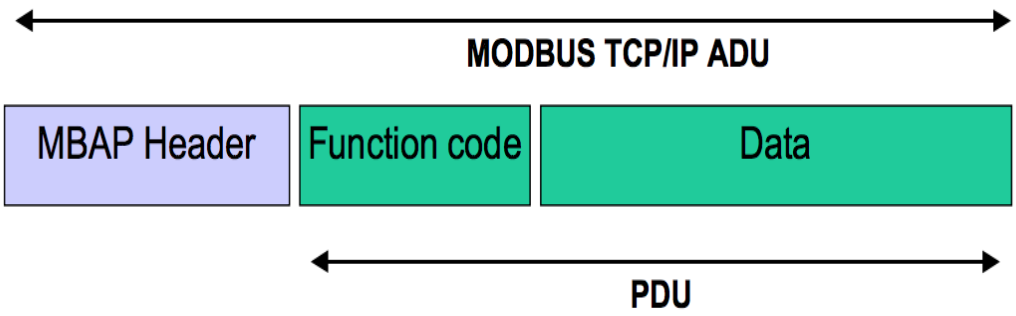


Figure 1. MODBUS request/response messages over TCP/IP. Adapted from [16].

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by the server from the received request

Figure 2. MODBUS Application Unit header description. Adapted from [16].

3. S7Comm

The S7 Communication Protocol, or S7Comm, is Siemens proprietary protocol for Siemens programmable logic controllers that uses the master/slave or client/server communication model through port 102 [17]. S7 header fields include information such as message type and data length. Data in S7 packets contains function code to differentiate between read and write jobs [18].

4. SNMP

The Simple Network Management Protocol (SNMP) is a network management protocol used to collect data from devices for management, configuration, or monitoring [19]. Although the latest version is SNMPv3, Conpot uses SNMPv2 over TCP on port number 161. The most common use for SNMP in industrial control systems is to manage and monitor a group of field devices from a master system by polling them on a regular basis [20].

5. BACnet

The Building Automation and Control Networking protocol (BACnet) is a proprietary communication protocol by ASHRAE which provides communications among different kinds of devices. According to [21], BACnet is commonly used for heating, ventilating, and air-conditioning control systems (HVAC), but also for many other industrial control systems. The Conpot default template uses port 47808 for BACnet.

6. IPMIs

The Intelligent Platform Management Interface (IPMI) is a protocol developed by Intel, Hewlett Packard, NEC, and Dell to manage hardware in a network through port 623 [22]. For industrial control systems, IPMI is primarily used to monitor data such as temperatures or power status of devices in the network. IPMI can also control devices by booting, restarting, or shutting them off. Documentation of device action and data are also logged using IPMI.

7. EtherNet/IP

The EtherNet Industrial Protocol (EtherNet/IP) was developed in the 1990s by the ControlNet International and Open DeviceNet Vendor Association (ODVA). It is built on top of Common Industrial Protocol (CIP) [23]. EtherNet/IP has explicit messages and implicit messages. Explicit messages are communication between client and server using port 44818, while implicit messages are communication between devices using port 2222. Conpot is limited to explicit messages since its purpose is to interact with attackers that initiate communications as clients.

8. CIP

CIP is an object-oriented protocol that is encapsulated in EtherNet/IP [18]. CIP represents data as objects [24]—an abstract representation of a vendor-specific component. CIP supports two types of objects: 1) common objects such as connection object, router object, and identity objects, and 2) device-specific objects. Although data

differ with the message type, packets always include an encapsulation header, which contains the command, length, status, and command-specific data [25].

D. INDICATORS OF COMPROMISE

Sanders and Smith [26] define indicators of compromise as attributes that indicate the presence of malicious activities. There are two main types, host and network indicators. Host-based indicators of compromise are extracted from host machines such as file hash, directory path, or file names while network-based indicators of compromise are data that can be found in communications over the network such as IPv4 or IPv6 addresses, URLs, or port numbers. An example is a well-known IPv4 address observed to be sending malware to different targets. Such indicators of compromise can be used to create a filter rule to prevent connections to the server. Indicators of compromise are commercially available or freely distributed by cybersecurity vendors and professionals. They are also available in the control-systems compartment of the US-CERT Secure Portal, but they are not as extensive as those for traditional systems [27]. Although indicators of compromise are effective proactively against attacks, they can only be extracted after the incidents have taken place. In this research, only network-based indicators of compromise were studied. Using Sanders and Smith's definition of indicators of compromise, we considered IP address, port numbers, the combination of IP address and port number, IP address locations, the frequency of communication, and size of data being transferred as examples of indicators of compromise. Indicators of compromise that may be unique to industrial control systems include protocol data such as MODBUS function codes, MODBUS slave identifications (ID), and EtherNet/IP command codes.

E. PREVIOUS WORK ON HONEYPOTS

Honeypots have been used for various kinds of cybersecurity research. Mokube and Adams [9] highlight honeypots as a new form of active defense for following reasons: Honeypots create a distraction from valuable assets, provide early warnings of an attack, and provide means of study during or after an attack. They outline that additional advantages of honeypots are:

1. Small dataset: Since honeypot data is only collected during an actual interaction, there is no false positive data and noise data is kept to a minimum.

2. Minimal resources: Honeypots do not require a large number of resources to implement.

3. Simple and flexible: Honeypots are relatively simple because they do not require new development, and can easily be configured.

4. New tools and tactics: Data from attacks can be used to discover new tools and tactics.

Mokube and Adams also discuss some disadvantages of honeypots:

1. Limited vision: Honeypots can collect information only during active attacks. For example, malicious activities on the host machine will not be recorded unless they involve the honeypot as well.

2. Identification: Experienced attackers may recognize the honeypot's behavior and identify it as a honeypot.

3. Takeover: A compromised honeypot may provide a new vector for the attacker to take over the entire system.

Yuksel, Hartog, and Etalle [28] discuss two types of detection mechanisms used by intrusion detection systems. Misuse-based detections focus on known attacks, greatly reducing false positive rates. However, this type of detection is ineffective against zero-day attacks. Anomaly-based detections identify anomalies by comparing the system behavior to that of a known baseline. This approach may be effective against unknown attacks but will produce high false positive rates. Yuksel et al. present a practical framework for anomaly-based detection by analyzing the syntactic and semantic interpretation of packet messages and protocol fields from real datasets captured from different industrial control systems. They conclude that although the framework is fairly successful, e.g., for MODBUS-TCP requests, the framework yielded the detection rate of

93.7% and no false positive rates. However, they emphasize that their framework heavily relies on the quality of the data parser and old data.

Redwood et al. focused on encouraging zero-day attacks or high-value attacks by using high-interaction honeypots such as GridPot and creating a framework that can detect anomalies successfully [29]. Due to the sensitivity of the study, details of the honeypot implementation were not discussed. However, the authors did mention they used a honeynet layer with GridPot, an interaction layer using real human-machine interface and emulated cyber-physical systems, an infrastructure modeling layer to generate realistic data, and a logging layer to collect and analyze data to alert the security professionals trying to defend their electric grid. Redwood et al. conducted multiple attacks on their system and concluded that their system was able to successfully recognize high-value attacks.

Piggin and Buffey [30] used a self-developed hardware honeypot configured to resemble an actual industrial control system. This study highlights that although software honeypots that are commonly used are easier to develop, deploy, and maintain, they are limited in that they cannot completely emulate a real industrial control system. Using real hardware to emulate a real-life industrial control system closely enables collection of detailed data on specific targeted attacks. These include password attacks, dictionary attacks, SSH brute force attacks, and attempts to execute malicious code. This study also observed heavy reconnaissance scanning which is similar to the study by Serbanescu et al. [31]. Their study concluded that although deploying a hardware honeypot on industrial control system is difficult and expensive, it allows cybersecurity professionals to better study and prepare for rapidly evolving trends in industrial control system attacks.

THIS PAGE INTENTIONALLY LEFT BLANK

III. PROBLEM DEFINITION AND ASSUMPTIONS

A. PROBLEM DEFINITION

In an ideal world, cybersecurity professionals for industrial control systems would be able to obtain indicators of compromise without financial, time, and manpower constraints. Even if they do, developing a hardware honeypot as Piggin and Buffey [30] did would allow the improvement of industrial control systems network security. However, both of these situations are unrealistic and difficult to achieve. As mentioned earlier, indicators of compromise are available commercially but they are mostly limited to information technology indicators of compromise rather than operational technology indicators of compromise. Even if abundant operational-technology indicators are available, most owners of systems will face financial restrictions to keep up with ever-changing indicators of compromise. Building a separate hardware honeypot platform to extract indicators would be difficult and time-consuming.

To address such problems, our study explored methods for extracting indicators of compromise from the attacker traffic information collected by Conpot. Conpot has limited manpower and financial requirements since it is an open-source honeypot and is fairly simple to install and use.

B. ASSUMPTIONS

For our study, we assumed that traffic to our honeypot would be light and its existence might require manual exposure via blog postings or classifieds advertisements. While some attackers might not recognize that our test system is an industrial-control-systems honeypot, we assumed that, using common reconnaissance tools, more advanced attackers would recognize it and would conduct operational-technology attacks.

C. SIMILAR WORK

A 2015 study by Serbanescu et al. [31] focuses on analyzing possible threats in industrial control systems using a self-developed and deployed large-scale low-interaction honeynet. The protocols used in their honeynet include MODBUS, DNP3,

ICCP, IEC-104, SNMP, TFTP, and XMPP. They observe that attackers used MODBUS, DNP3, and SNMP more often than other protocols on their honeynet. Other observations include heavy reconnaissance activities, such as port scanning and a lack of specific targeted attacks. These results suggested using high-interaction honeypots to evaluate the possible threats in industrial control system.

Research by Kuman et al. was similar to the experiment conducted by Redwood et al. in that a honeynet was used to create an intrusion-detection system [32]. Kuman et al. used Conpot and IMUNES, a network emulator, to create a honeynet and host-based intrusion-detection system by observing the changes made in Conpot activity logs. The study concluded that changes in the log resulting from attacker activity on Conpot were detected by the intrusion-detection system even if the activities were as simple as a port scan.

IV. METHODOLOGY

A. SETUP

Our study used a Dell laptop computer with a Linux 64-bit Ubuntu 16.04.3 LTS operating system with 16 GB of RAM and 750 GB hard disk. A virtual machine was installed using Oracle VM Virtualbox 5.1.20. Conpot was installed and deployed in this virtual machine instead of the actual host machine. The Conpot virtual machine also ran a Linux 64-bit Ubuntu 16.04.4 LTS operating system that was configured with 2048 MB of RAM and expandable hard disk memory space up to 10 GB of 750 GB. Our experiment network was set up outside of Naval Postgraduate School's firewall to make it easier for attackers to discover the honeypot. Both the host and virtual machines use statically assigned IPv4 addresses x.x.x.54 and x.x.x.55. Both machines use internal bridged networking for communications. Figure 3 shows the experimental network setup.

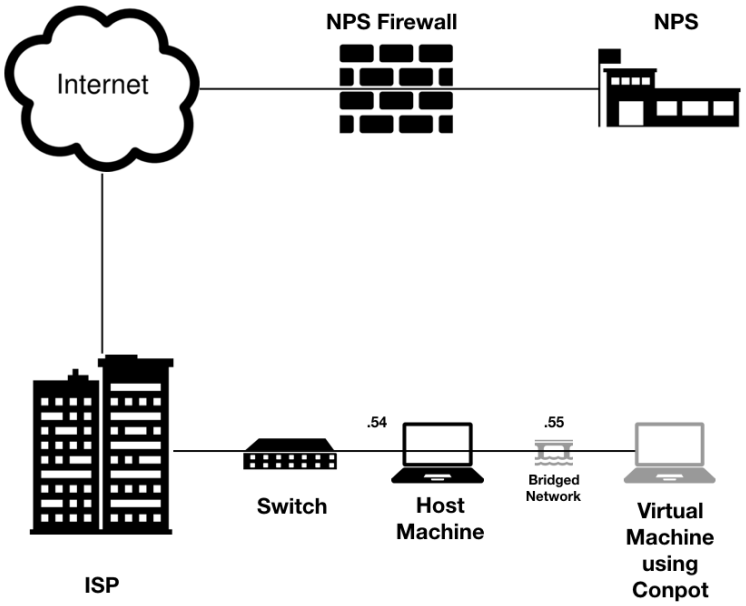


Figure 3. Experiment Network Setup

Conpot version 0.5.1 was installed in the virtual machine and deployed on a .55 network using the Conpot-provided default template. The default template is configured to simulate an electric-power plant using Siemens SIMATIC S7-200 programmable logic controllers along with other industrial control system devices. Seven protocols and their port numbers used in the default template are listed in Table 1. All seven protocols were configured to use the IP addresses of the Conpot network. Conpot was launched through the Linux terminal by executing the command line "`sudo conpot --template default`." Detailed Conpot installation steps are provided in the appendix, Conpot Installation.

Table 1. Protocols used by Conpot's default template

Protocol	Port number
HTTP	80
EtherNet/IP	44818
MODBUS	502
S7Comm	102
SNMP	161
BACnet	47808
IPMI	623

B. DATA COLLECTION

Our honeypot collected data over four months from October 4, 2017, to February 15, 2018. Conpot ran during this time period continuously except when its log had to be saved and backed up. When that occurred, Conpot was launched again using the default template. The network protocol analyzer Wireshark [33] was used to monitor and capture network traffic. Table 1 gives the timeline of data collection.

Table 2. Data collection timeline

Date	Action
2017-10-04 ~ 2017–10-04	• Tested deployment of Conpot after installation
2017-10-06 ~ 2017–10-10	• Brought Conpot online on 2017–10-06. • Took Conpot offline on 2017–10-10 to collect conpot.log and back up data.
2017-10-10 ~ 2017–10-16	• Brought Conpot online on 2017–10-10. • Took Conpot offline on 2017–10-16 to collect conpot.log and back up data.
2017-10-16 ~ 2017–10-20	• Brought Conpot online on 2017–10-16. • Took Conpot offline on 2017–10-20 to collect conpot.log and back up data.
2017-10-20 ~ 2017–11-02	• Brought Conpot online on 2017–10-20. • Took Conpot offline on 2017–11-02 to collect conpot.log and back up data.
2017-11-02 ~ 2017–11-16	• Brought Conpot online on 2017–11-02. • Took Conpot offline on 2017–11-16 to collect conpot.log and back up data.
2017-11-16 ~ 2017–11-27	• Brought Conpot online on 2017–11-16. • Noticed increased EtherNet/IP traffic. • Started packet capture of Ethernet/IP traffic (port 44818). • Took Conpot offline on 11–27-2017 to collect Conpot.log and back up data.
2017-11-27 ~ 2017–12-07	• Brought Conpot online on 2017–11-27. • Continued capturing Ethernet/IP traffic with Wireshark. • Noticed increased MODBUS traffic. • Started packet capture of MODBUS traffic (port 502). • Took Conpot offline on 2017–12-07 to collect conpot.log and back up data.
2017-12-07 ~ 2018–01-08	• Brought Conpot online on 2017–12-07. • Continued packet capture of Ethernet/IP and MODBUS traffic. • Conpot abruptly stopped logging on 2017–12-13 after a new MODBUS connection was established with the following IP address and TCP port pair: 172.104.245.71:56042. • When Conpot was stopped on 01–08-2018, its logging mechanism resumed and a number of activities were logged as if they occurred in 01–08-2018. These activities might have occurred before this date. • Took Conpot offline on 2018–01-08 to collect conpot.log and back up data.
2018-01-08~ 2018–01-11	• Brought Conpot online on 2018–01-08. • Continued packet capture of Ethernet/IP and MODBUS traffic. • Took Conpot offline on 2018–01-11 to collect conpot.log and back up data.

Date	Action
2018-01-11 ~ 2018–01-18	• Brought Conpot online on 2018–01-11. • Continued packet capture of Ethernet/IP and MODBUS traffic. • Conpot log abruptly stopped logging on 2018–01-13 after a new MODBUS connection was made with IP/port pair 172.104.250.171:34954. • When Conpot was stopped on 2018–01-18, remaining data in cache was dumped and logged into conpot.log as activities occurred in 01–13-2018. These activities might have occurred before this date. • Took Conpot offline on 2018–01-18 to collect conpot.log and back up data.
2018-01-18 ~ 2018–01-24	• Brought Conpot online on 2018–01-18. • Continued packet capture of EtherNet/IP and MODBUS. • Noticed IPMI traffic increased before Conpot logging was abruptly interrupted. Started capturing IPMI packets (port 623). • Captured all processes running initially when Conpot was deployed on 2018–01-18. • Took Conpot offline on 2018–01-11 to collect conpot.log and back up data. • VM failed to expand disk space on 2018–01-24 and this issue was discovered on 2018–01-29. Due to lack of disk space, any activity after 2018–01-24 on conpot.log was lost and all packet capture of EtherNet/IP, MODBUS, and IPMI after 2018–01-18 was lost.
2018-02-12 ~ 2018–02-15	• Freed up 2.7 GB of memory by purging old kernels and orphaned packages. • Brought Conpot online on 2018–02-12. • Took Conpot offline on 2018–02-15 to collect conpot.log and back up data.

C. PARSER DESIGN

To analyze logs specific to each protocol, we wrote the parsers in Python. These used regular expression to filter out desired information from *conpot.log* and displayed the remainder. There were six protocol-specific parsers and an overall parser. Data in Conpot's log file (*conpot.log*) showed that most of the log entries only contained limited flow data between two points. Flow data is simply a log format that includes, but not limited to, source and destination IP address and port combination. Flow may also include other protocol relevant information. In our study, captured flow data included timestamps, IP addresses, ports, and other protocol-specific options used between the source and

destination. This limited our parser design to concentrate on the extraction of data that we can gather from flow data.

Each parser implemented both general and protocol-specific functions. General functions included calculating statistics on protocol activities by date, IP addresses, port information, geolocation of IP addresses, and request and response information that may provide signatures to create indicators of compromise. All parsers employed the Pandas Python package to plot statistical data, Geolite2 Python package to map IP addresses to geographical locations, and Pickle package to export data as comma separated value files. Since no SNMP activity was present in our data, an SNMP parser was not written.

Conpot records all HTTP requests received and responses sent out by Conpot. The HTTP parser filters out HTTP requests and categorizes them based on the HTTP version, access methods, and date of activity. These data are used to determine the distribution of the versions and methods used by the attackers, and to calculate the HTTP-request activity count and the country activity count, both by date. Finally, a list of countries is organized to sum up the activity count of each country. HTTP request messages are ignored due to the limitation of our parser. Even though there are much valuable information contained in HTTP request messages, our parser was not sophisticated enough to extract and organize the contents of HTTP request messages efficiently.

For MODBUS protocol activity, Conpot logs two different types of flow data: connections and traffic. Conpot records connection flow data when an attacker establishes a successful TCP connection with Conpot using port 502. It captures traffic flow data when an attacker attempts to send a payload over TCP to Conpot using port 502. The connection flow data stored in the log file only provide the timestamp, IP address, and port number of the source (attacker). The traffic flow data provide the timestamp, source IP address, function code, and slave ID used by the attacker. The MODBUS parser filters both connection and traffic activities from *conpot.log* to keep the total MODBUS activity count by date and country activity count by date. Then, they are sorted into two different hash tables, one to store connection flow data and the other for traffic flow data. Both hash tables use the source IP address as the key. The connection hash table only contains the source port numbers. The values kept in the traffic hash table are the source port numbers,

function codes, and slave IDs. From the traffic hash table, the MODBUS parser uses the function codes and slave IDs to produce a distribution of the different types of MODBUS requests since those two fields specify which requests are being made.

Conpot records all EtherNet/IP traffic it generates. The EtherNet/IP parser filters out all the requests to extract command code, port, and date information and stores them in a hash table using the IP address as the key. The country of origin is also added as a value to this hash table. The parser uses the hash table to calculate the distribution of the command codes used by the attackers,\ and to gather the request activity count and country activity count by date. The parser also sums up the country counts to obtain the total country activity distribution.

For S7Comm protocol activity, Conpot maintains three different types of flow data: session, connection, and packet. Conpot keeps track of the session flow by recording the source IP address and assigning a unique session ID for that address when an initial TCP connection request to Conpot's port 102 is made.

When a successful TCP connection for S7Comm messages is established, Conpot generates a connection flow data record that lists the source IP address, port number, and session ID that was assigned to that source IP. When a source IP address establishes multiple connections to Conpot within a short amount of time using multiple source ports, Conpot will assign the same session ID to all connections since they all originate from the same source IP address that originally made the initial connection request. However, if one source IP address makes multiple connection requests over a longer period, new session IDs will be generated for each connections requests.

If a source sends an S7Comm payload to Conpot, a payload flow data record is created, which contains information such as session ID, protocol data unit (PDU) type, request ID, data length, and request ID. The PDU type indicates what message is contained in the packet. For example, PDU type 1 means that the client is sending a "job request" message and PDU type 7 means that the client is sending "User data" needed for debugging and programming purposes [17].

The S7Comm parser filters out information about the S7Comm connections, sessions, and packets, and stores them in three separate lists. All three lists are iterated to obtain the activity count and country activity count by date. The total country distribution is calculated by summing up the country activity counts of all three flow data. From the list containing the session flow data, the IP addresses and the corresponding session IDs are extracted and stored in a hash table using the IP address as the key and the session ID and country of origin as values. These session IDs are also used as keys to look up session-specific data in a nested hash table. From the connections list, port numbers, PDU type, data length, and requests IDs are extracted and stored in the same hash table by matching up the session IDs in the nested hash table. The completed hash table is used to generate an overall statistic to determine the trend of the PDU types and data length being used. The request IDs are also collected to see if we can identify any patterns from them.

For BACnet activity, Conpot keeps tracks of four different types of flow data: session, connection, PDU, and decoding error. Much like the way it handles the S7Comm protocol, Conpot creates a session flow record when an initial connection request is made. Once a connection is established, connection flow record is generated. The PDU flow record is generated when Conpot receives a BACnet PDU. The PDU type indicates which PDU type is being used. Examples of PDU type include “BACnet-confirmed-request” and “BACnet-unconfirmed-request” [34]. Decoding error messages are recorded by Conpot when an invalid PDU type is sent. Information in the PDU flow record and the decoding error messages are not distinguishable to serve as a unique identifier.

The BACnet parser keeps the information relating to the session, connections, PDU, and decoding error flow data in four different lists. Only the sessions, connections and PDU lists are iterated to gather the total activity count by date. The decoding error message list is not included in the activity count since those error messages are generated by Conpot, not by the attacker. Since both the PDU and decoding error flow records do not provide any information that can serve as a unique identifier, only the session and connections lists were used to generate a hash table that is used to identify the overall BACnet activity trends using the IP address as the key. This hash table only has the session

ID, which also serves as a key to a nested hash table. The values kept in the nested hash table are the port numbers, countries and access dates.

Conpot's IPMI data is the most limited out of all protocols. There are four different types of flow data: traffic, session, incoming traffic, and closed session. The traffic, session, and incoming traffic flow data consist of the timestamp, source IP address, and source port. The closed session flow data only provide the timestamp and source IP address. When a new connection request is made to Conpot's port 623, Conpot generates a traffic flow record consisting of the timestamp, source IP address and port number. When this connection is established, Conpot generates a session flow record that includes the timestamp, source IP address and port number. When a connection is successfully established, Conpot generates a session flow record immediately after generating the traffic flow record. However, when the connection is not fully established, only the traffic flow record is generated. When an attacker makes another connection using the same IP address and port number combination, Conpot generates an incoming traffic flow record to distinguish it as a returning traffic. When a connection from an attacker is terminated, a closed session flow data record is generated.

The IPMI parser keeps track of the IPMI flow data and stores them into a list. The parser iterates through the list to extract the date, source IP address, and port number. It then checks if the traffic is a returning traffic or if the session is closed. Using these data, four different hash tables are formed. One hash table uses the date as the key and stores the sum of activity counts as its value. The second hash table uses the date as its key and stores the activity count by country as its value. The third hash table uses the country as its key and stores the total activity count as its value. The fourth hash table uses the source IP address as its key and stores the date, port numbers, whether or not the traffic is a returning traffic, and whether or not that session is closed as its value. The last hash table value is used for characterizing the overall trend of IPMI data.

Exported CSV files from all six protocol-specific parsers are used as an input the overall parser. Using the data contained in these CSV files, the overall parser generates the overall statistics on protocol activities, country count, protocol distribution of individual countries, and activity count by date by individual countries.

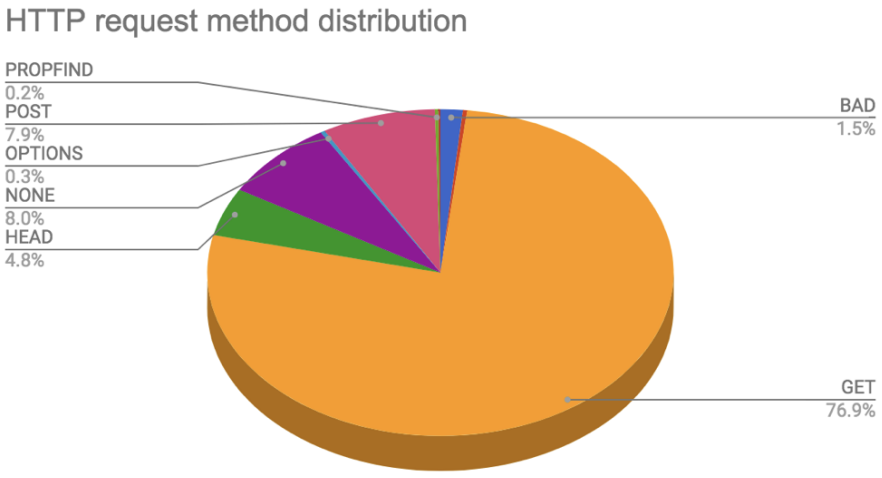
V. DATA AND RESULTS

From 2017–10-04 to 2018–02-15, Conpot recorded 38,472 log entries (4.5 MB). Packet captures of EtherNet/IP and MODBUS traffic were made (totaling 1.3 MB) over seven different sessions. This chapter analyzes the data collected for each protocol used by the attackers and the obstacles encountered.

A. PROTOCOL DATA AND RESULTS

1. HTTP

Conpot reported a total of 7,366 HTTP requests and responses. HTTP version 1.1 was the most prominent with a count of 6,214. HTTP version 0.9 had the second highest count of 652 and HTTP version 1.0 had 498. There were two HTTP version 0.0 attempts from the United Kingdom. Seven different HTTP methods were used along with HTTP requests that did not use any methods (shown as "None") or sent an invalid HTTP version 0.9 request. Figure 4 shows the distribution of HTTP methods.



	Bad	CONNECT	GET	HEAD	None	OPTIONS	POST	PROPFIND	PUT
Count	57	11	2840	176	294	11	292	9	5

Figure 4. HTTP request method distribution

The observed HTTP activities came from 53 countries, with the top three being the United States, China, and Brazil in order. HTTP activity was significant around October and November of 2017 and gradually declined from the end of November of 2017. October 24th of 2017 showed the highest activity of 401 counts with 368 attempts originating from the United States and the rest from Nepal, Russia, Italy, Colombia, Germany, Brazil, China, Hong Kong, Indonesia, India, and the Republic of Korea (Table 3).

Table 3. Country activity count for HTTP on 2017–10-24.

Country	Count
United States	368
China	9
Italy	6
Republic of Korea	4
India	4
Hong Kong	3
Brazil	2
Columbia	1
Indonesia	1
Russia	1
Germany	1
Nepal	1

2. MODBUS

There were 2,316 MODBUS activities involving seven countries with the United States producing the most activities (Figure 5). Only three of the nineteen function codes were used: 0x03 (Read Holding Register), 0x2b (Read Device Identification), and 0x11 (Report Server ID). Function code 0x2b had the highest count of 49. Only 59 used a valid MODBUS function code. From this, we can assume that the attackers were simply

making scanning attempts rather than performing specific attacks using MODBUS protocol.

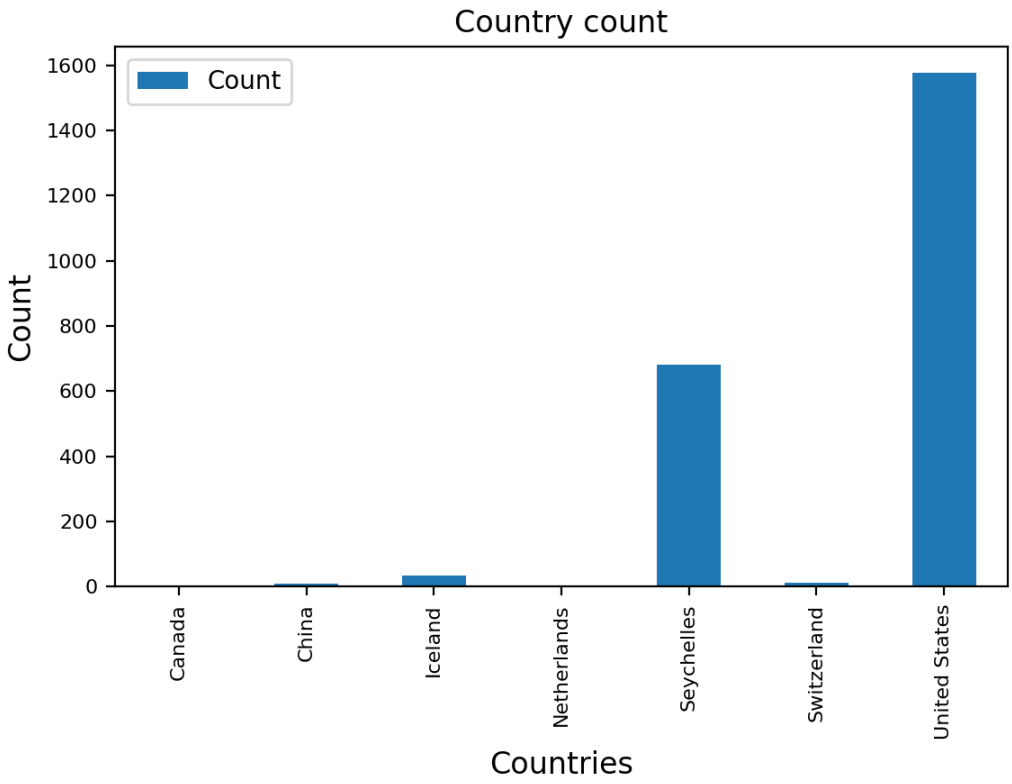


Figure 5. MODBUS-activity country distribution

Other interesting events were spikes of activity counts on November 13, November 23, December 8 of 2017, and January 10 of 2018 (Figure 6). All four dates showed that the activities originated from the United States and Seychelles made multiple requests using function codes 0x11 and 0x2b originating from multiple ports and slave IDs. For example, one attacker from the United States made multiple requests using slave IDs from 0 to 255, port numbers from 34617 to 40881, and function codes of 0x11 and 0x2b on November 13, 2017. On November 16, 2017, ICS-CERT released a security advisory [35] that described the vulnerabilities associated with Siemens SICAM equipment, and MODBUS is a protocol supported by this product. This date is fairly

close to our first and second activity spikes, and the equipment emulated by Conpot is also a Siemens product.

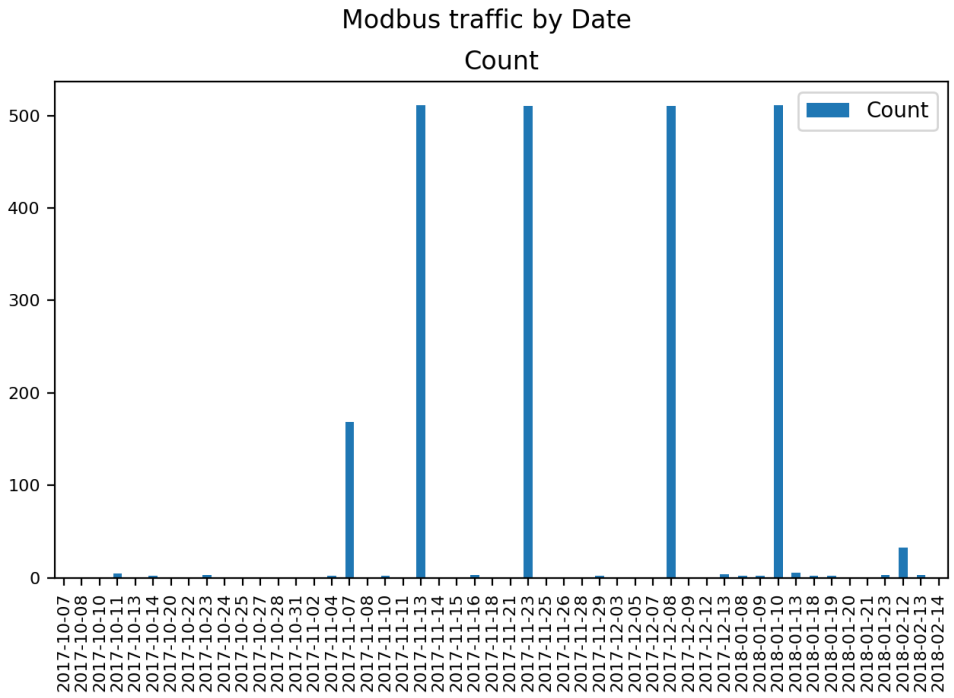
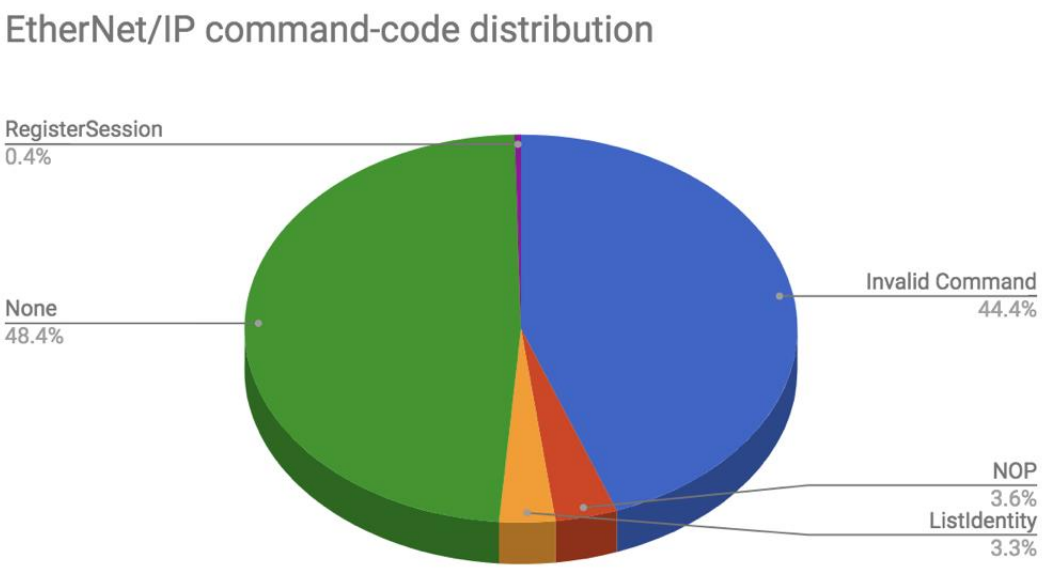


Figure 6. MODBUS traffic over time

3. EtherNet/IP

Out of 154 counts of EtherNet/IP activities, only NOP (No Operation), Register Session, and List Identity commands were used. Invalid command or null command were also observed (Figure 7). Most days only had one attempt except for November 22, 25, 27 of 2017, when there were 28, 34, 28 activities, respectively. On January 19, 2018, 34 activities were recorded. The traffic on these dates originated from the United States except for November 27, 2017, on which the attacks came from China.



	Invalid command	NOP	ListIdentity	None	RegisterSession
Count	122	10	9	133	1

Figure 7. EtherNet/IP command-code distribution

4. S7Comm

There were 645 S7Comm activities captured in the log file. Out of 645, 436 connections were made, 60 sessions were established, and 149 packets were sent. All packets were sent with a PDU type of 1 or 7, a data length of 0 or 8, and a request ID of 0. According to [17], PDU type 1 indicate job request message type and PDU type 7 indicates user data message type. Japan showed the highest number of S7Comm activity followed by the United States (Figure 8). S7Comm activities from Japan occurred almost daily throughout the data collection period while attacks using other protocols were rarely made from Japan.

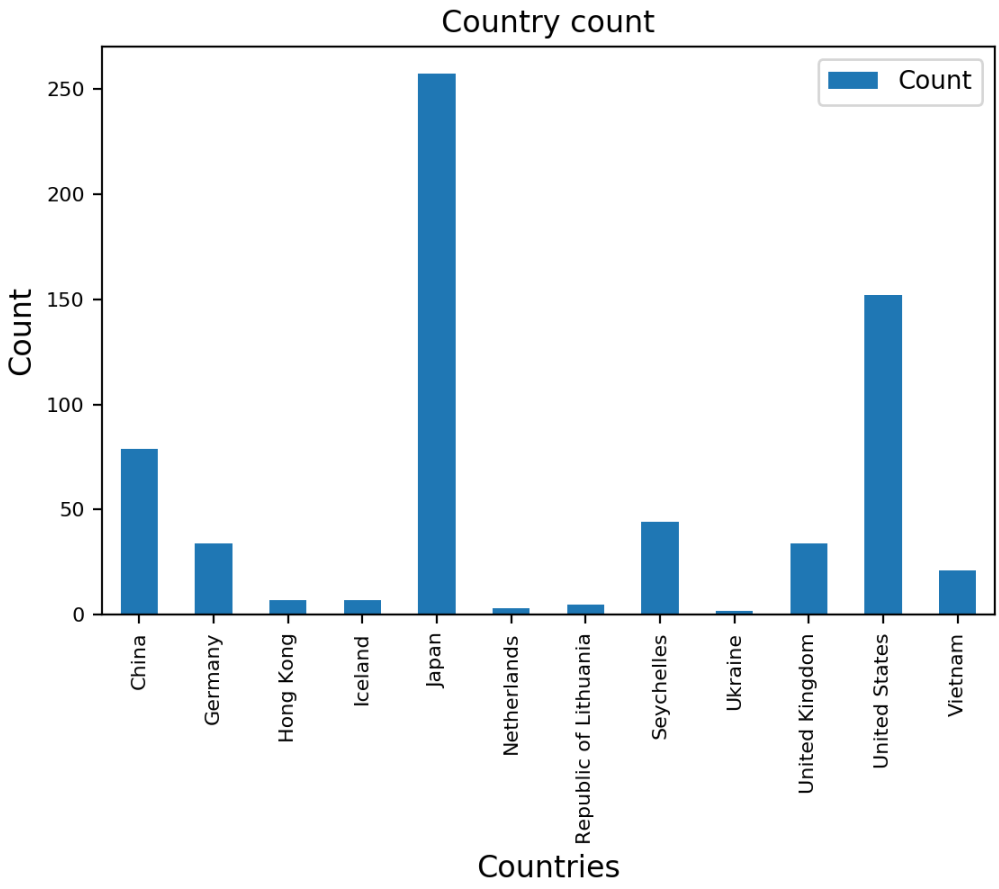
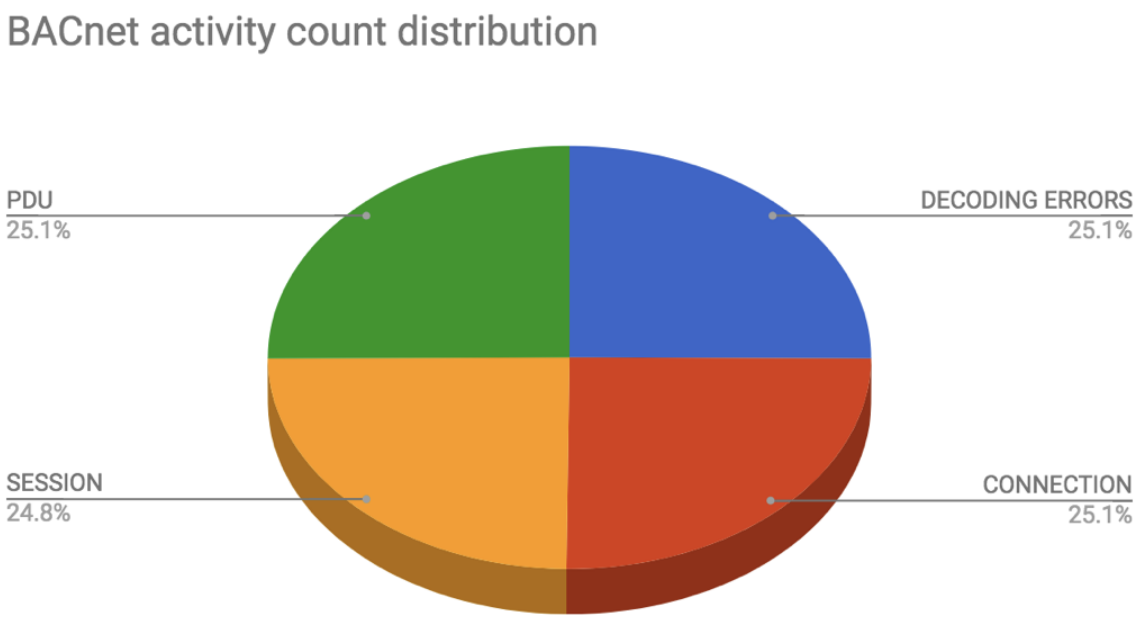


Figure 8. S7Comm country distribution

5. BACnet

The BACnet log showed 311 activities. Figure 9 shows the distribution of activity counts. As explained earlier in our BACnet parser design, Conpot did not provide any information that can be used as unique identifiers for PDU and decoding error message flow data. From Figure 9, we hypothesize that some data were sent for all 78 established connections but all of them used an invalid PDU type, resulting in 78 decoding errors. It is only a hypothesis because we have no way of identifying from where the PDU or decoding error flow data originated. Activity counts alternated between two and four, mostly originating from Switzerland, and the United States. January 8, 2018, particularly

stood out with 33 activity counts from Seychelles, Singapore, China, Switzerland, United States, and Canada.

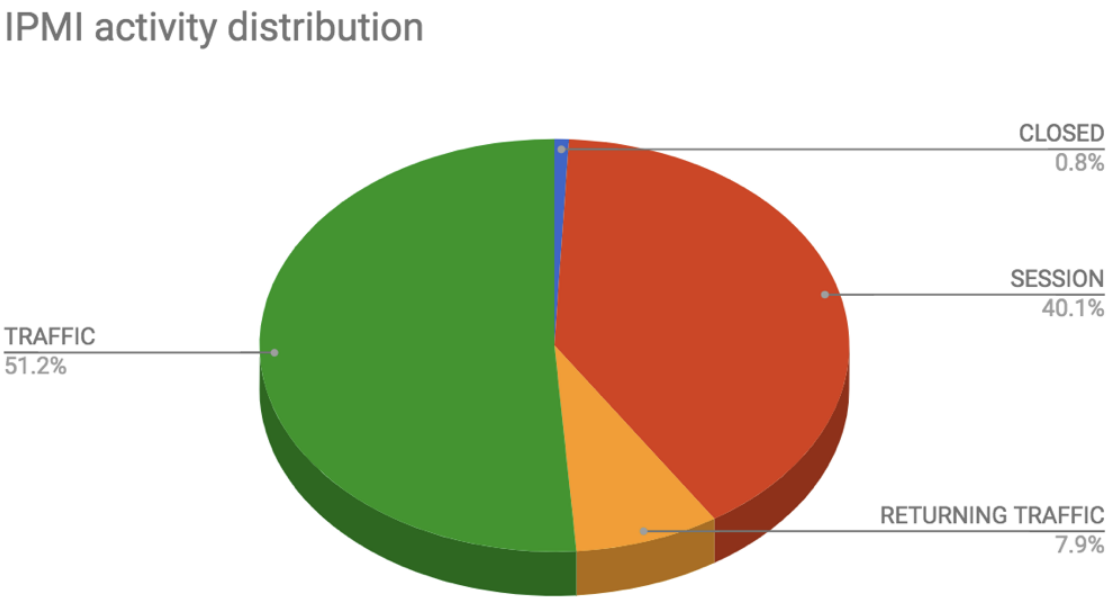


	Decoding errors	Connection	Session	PDU
Count	78	78	77	78

Figure 9. BACnet activity count distribution

6. IPMI

As shown in Figure 10, of 262 IPMI activities, 101 sessions were established, 129 new traffic was present, 30 activities were returning traffic, and only two sessions were properly closed. The United States was most active (233 activities) and Seychelles was the second most active (15 activities). As noted in our IPMI parser design, Conpot provided very limited data for IPMI activities and no other protocol-specific information could be extracted.



	Closed	Session	Returning traffic	Traffic
Count	2	101	30	129

Figure 10. IPMI activity distribution

7. Overall Statistics

As shown in Figure 11, a higher number of overall protocol activities were present in the month of October and November of 2017 and the number gradually declined over the data collection period. However, Figure 10 shows a few surges of activities due to high MODBUS activities discussed earlier. The statistics of all protocols (Figure 12) indicate that HTTP, MODBUS, and S7Comm were more popular, with 50.58%, 31.70%, and 8.83% of the activities respectively. Figure 13 shows that HTTP traffic was heavy at the beginning and slowed down over time. The other five protocols were more steady except for a few spikes.

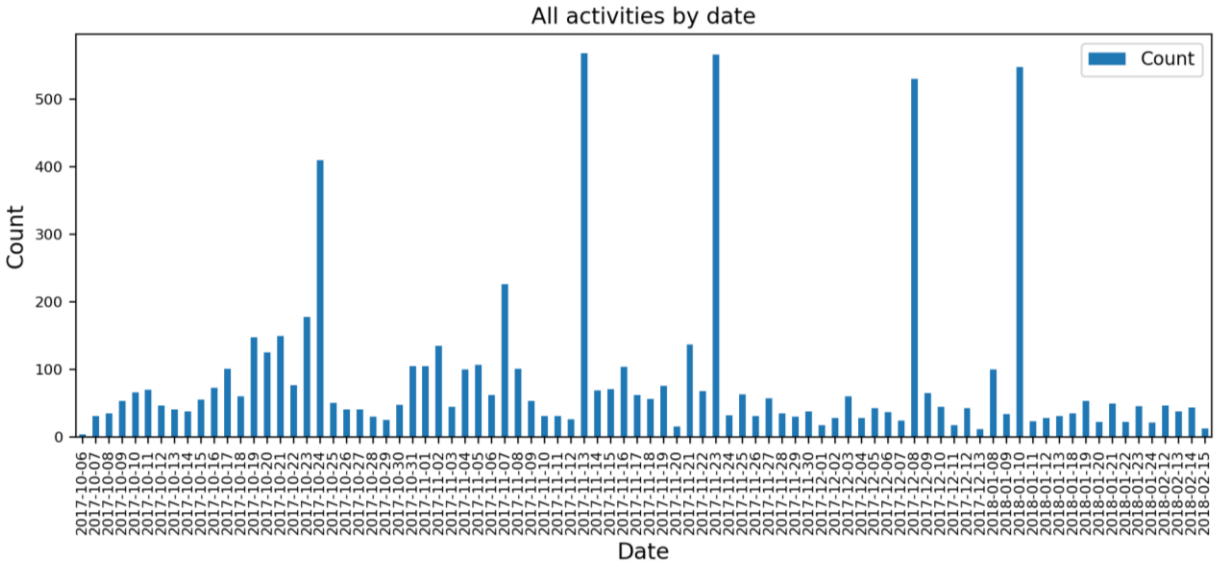
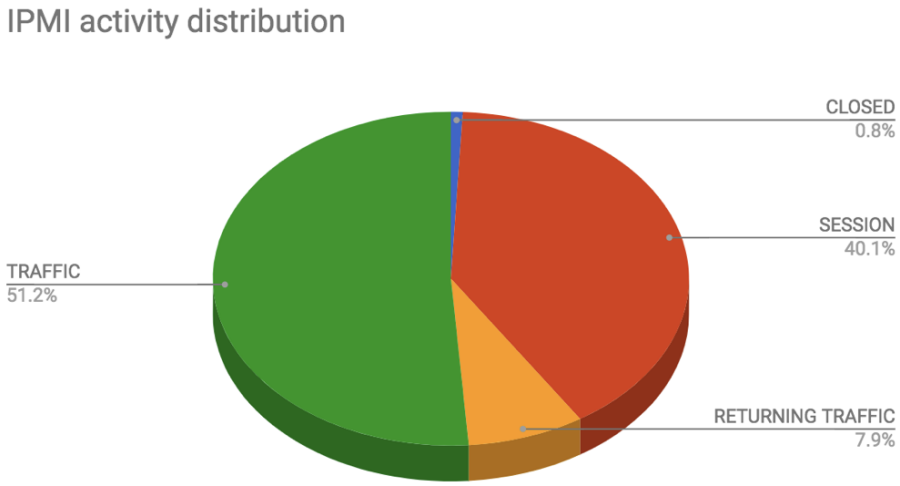


Figure 11. Activity count over time of all protocols



	Bacnet	ENIP	HTTP	IPMI	S7Comm	modbus
Count	233	154	3695	262	645	2316

Figure 12. Overall protocol distribution

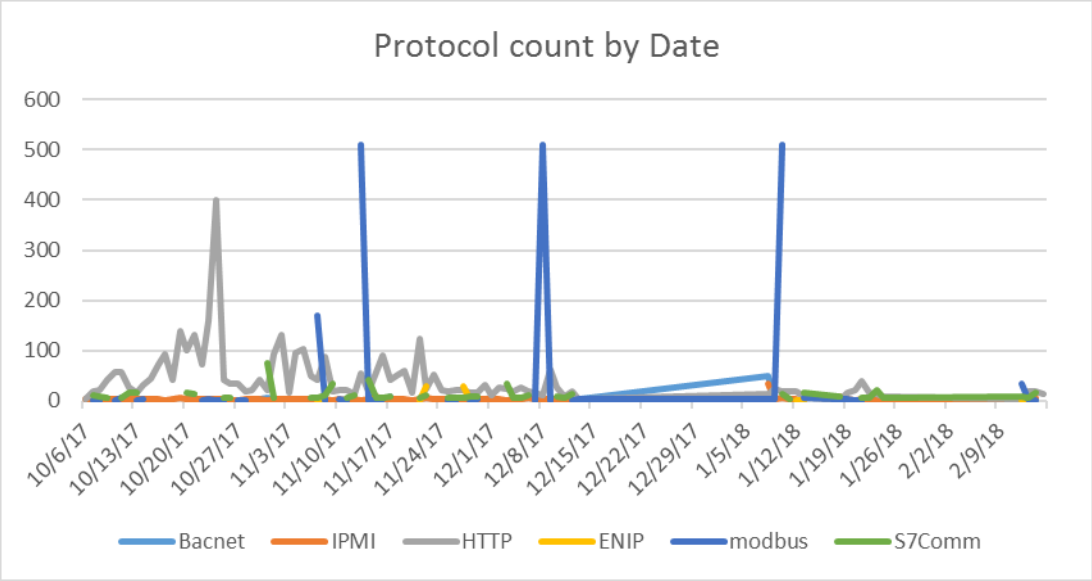


Figure 13. Protocol counts over time

Our Conpot was attacked by 54 countries, with the average of 10.51 attacks per day. Most of the countries focused on attacking the HTTP protocol. Attackers from the United States, Iceland, and Seychelles had the highest number of activities on MODBUS protocols, while attackers from Japan primarily attacked the S7Comm protocol (Table 4).

Table 4. Protocol activity distribution for USA, Seychelles, Iceland, and Japan

Column1	United States	Seychelles	Iceland	Japan
BACnet	84	10		
ENIP	111	3		
HTTP	982	42		126
IPMI	233	15	2	
Modbus	1578	680	33	
S7Comm	152	44	7	257

8. Outliers

After a high number of MODBUS attacks on December 13, 2017, Conpot abruptly stopped responding to any activities and became incapable of logging any information. There were also a high number of MODBUS attacks on November 13, 2017. When Conpot was stopped on January 8, 2018, it crashed while updating the log file. We assume Conpot was flushing the leftover log entries saved in its cache when it halted. Due to this event, other information that could have been collected was lost. The same failure happened again on January 13, 2018. Since November 13 also experienced a high volume of attacks, we hoped to see the same phenomenon happen on February 13, 2018, but it did not.

Due to Conpot's design that limited data collection to flow data, many protocol-specific information we wished to extract were absent. To mitigate the lack of information, full packet capture using Wireshark was conducted for EtherNet/IP and MODBUS protocols by capturing traffic on ports 44818 and 502. The EtherNet/IP capture was taken from November 22, 2017, to January 18, 2018, divided up into four sessions and MODBUS from November 30, 2017, to January 18, 2018 ,divided up into three different sessions. Full packet capture data was stopped after January 18 to make space for the Conpot log.

Most of the MODBUS protocol header information in the MODBUS packet capture matched the flow data recorded in the Conpot log file and did not provide any additional information. However, one packet captured on December 13, 2018 contained the string `objectClass0` in the hexadecimal dump of its payload and a function code of 0x01 (Read coils) as shown in Figure 14 and Figure 15. This contradicts our earlier observation of only seeing function codes 0x03, 0x11, and 0x2b in MODBUS activities. When *conpot.log* was manually searched using the IP address, the log failed to show this particular activity. Judging from the date, we assumed that this is one of many activities missed by Conpot during the mysterious crash occurred on December 13, 2017.

The EtherNet/IP packet captures also contained protocol header information that matched the flow data obtained from our Conpot log file—no additional information was

found. However, we noticed that attackers attempted to send non-EtherNet/IP packets over the TCP port reserved for EtherNet/IP, i.e., port 44818. Figure 16 and Figure 17 show an example of a TCP packet containing Microsoft Networks SMB protocol information in its payload while Figure 18 and Figure 19 show an HTTP/1.0 Get request. Our Conpot log showed that the corresponding packets were sent with "None" as the EtherNet/IP command code.

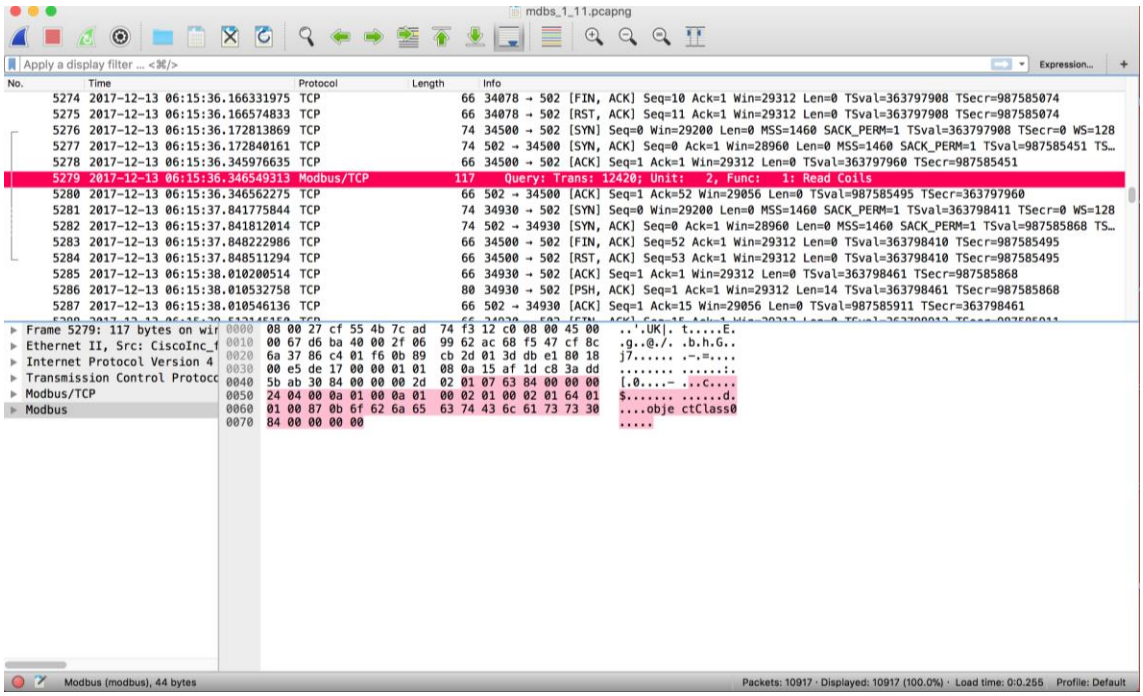


Figure 14. Frame 5279: Hex dump of suspicious MODBUS packet payload.

```
74 f3 12 c0 08 00 45 28   ..'.UK|. t.....E(
de d7 ac 68 fa ab cf 8c   .g..@.0. ...h....
60 33 9f 31 a0 17 80 18   j7...... `3.1....
08 0a 12 f9 ea b7 fa 49   ........ .......I
02 01 07 63 84 00 00 00   .W0....- ...c....
00 02 01 00 02 01 64 01   $....... ......d.
63 74 43 6c 61 73 73 30   ....obje ctClass0
                          .....
```

Figure 15. Enlarged Hex dump of Figure 13.

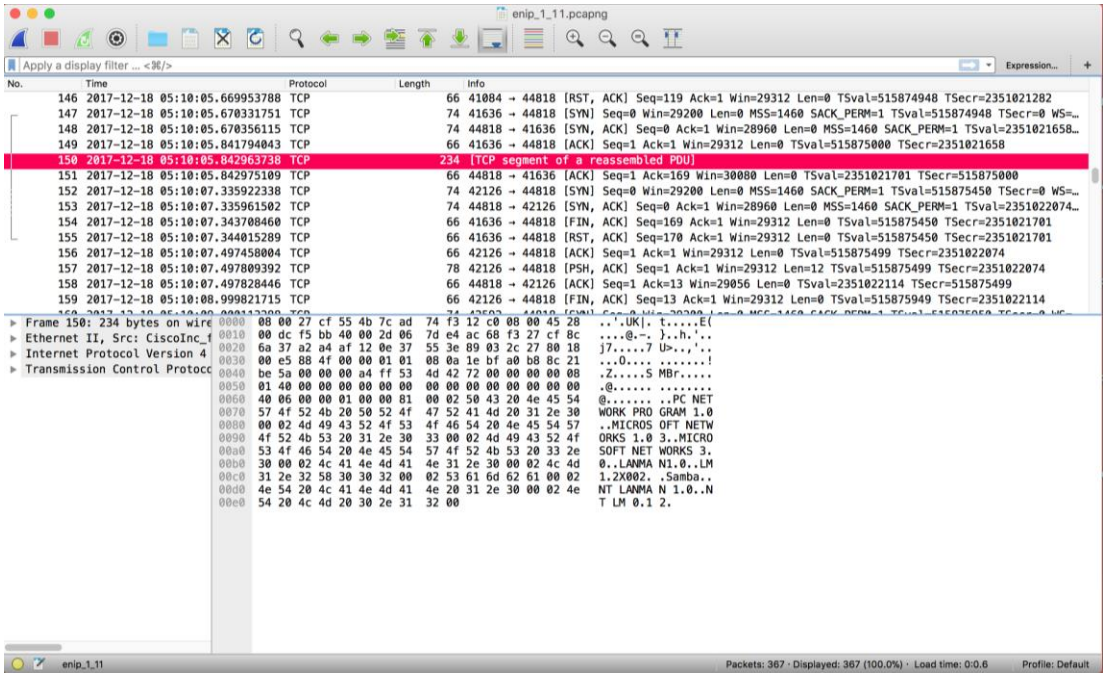


Figure 16. Frame 150: Hex dump of a TCP packet sent to port 44818 with Microsoft SMB protocol information.

```
2 c0 08 00 45 28   ..'.UK|. t.....E(
c 68 f3 27 cf 8c   ....@.-. }..h.'..
9 03 2c 27 80 18   j7.....7 U>..,'..
e bf a0 b8 8c 21   ...O.... .......!
2 00 00 00 00 08   .Z.....S MBr.....
0 00 00 00 00 00   .@...... ........
0 43 20 4e 45 54   @....... ..PC NET
1 4d 20 31 2e 30   WORK PRO GRAM 1.0
4 20 4e 45 54 57   ..MICROS OFT NETW
2 4d 49 43 52 4f   ORKS 1.0 3..MICRO
2 4b 53 20 33 2e   SOFT NET WORKS 3.
e 30 00 02 4c 4d   0..LANMA N1.0..LM
1 6d 62 61 00 02   1.2X002. .Samba..
1 2e 30 00 02 4e   NT LANMA N 1.0..N
                   T LM 0.1 2.
```

Figure 17. Enlarged Hex dump of Figure 15.

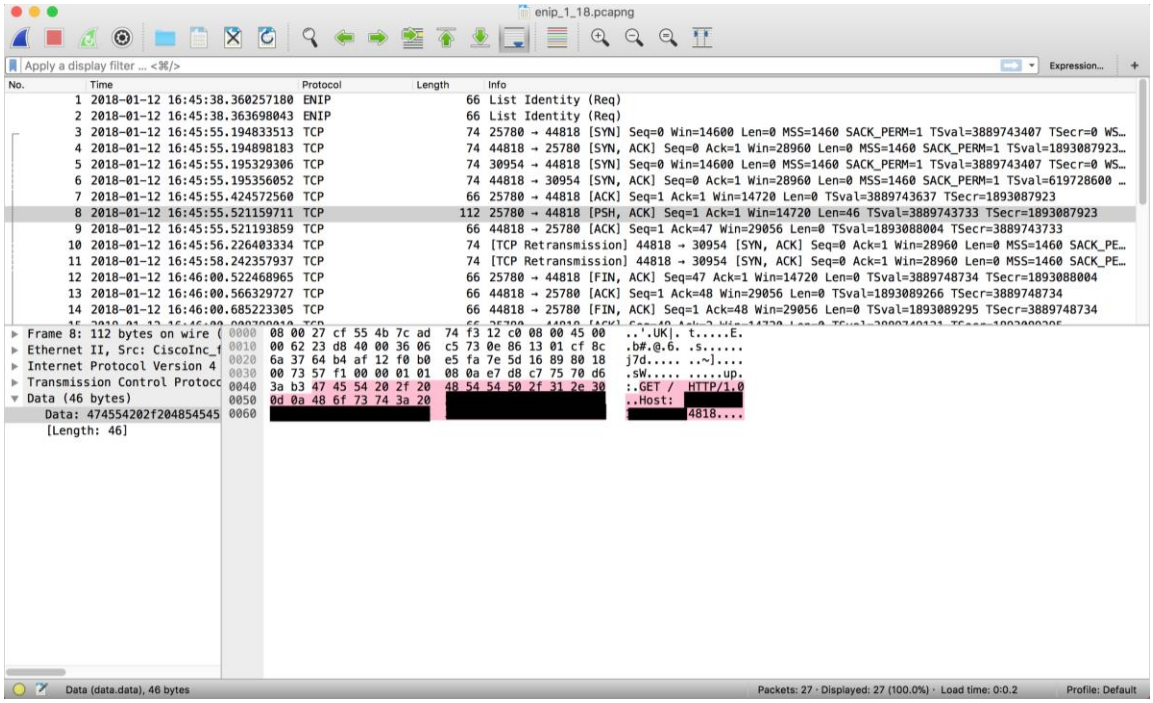


Figure 18. Frame 8: TCP packet (sent to port 44818) payload hex dump showing HTTP/1.0 get request

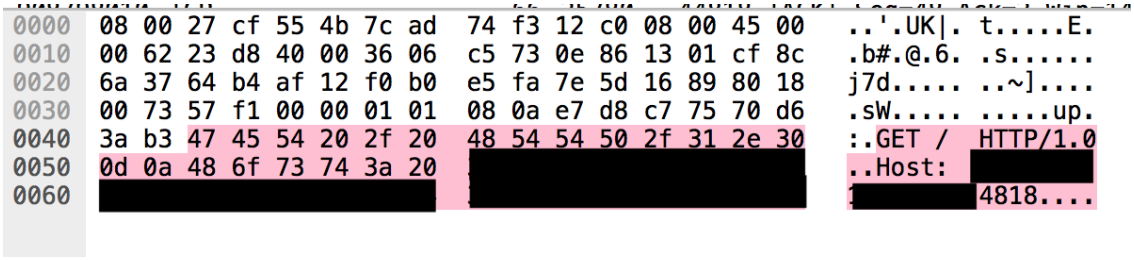


Figure 19. Enlarged Hex dump of Figure 17.

B. DIFFICULTIES

The most difficult challenge in our data collection was the limitation on hard disk space in our virtual machine. Although it was initially set to 10GB with expandable storage space, VirtualBox failed to expand the storage space due to a virtual-machine file-format error. This caused the loss of all data between January 24, 2018 and February

12, 2018. When we noticed the heavy volume of MODBUS traffic on the 13th of every month, a process dump of the virtual machine was performed before taking Conpot online on January 18, 2018. We intended to capture another process dump of the virtual machine after February 13, 2018, to find suspicious processes, but the virtual machine ran out of disk space and taking a process dump was not possible. After we managed to free up some space in our virtual machine and Conpot was taken online again on February 12, 2018, and a process dump was not captured to conserve the disk space for *conpot.log*. Conpot did not crash or experience high MODBUS traffic afterward.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

The goal of this research was to test if Conpot is a good method to extract data of malicious activity. Our study collected data using seven industrial-control-system protocols using Conpot. As shown in Figure 10, HTTP protocol generated the most traffic in Conpot rather than other protocols used by industrial control systems. From this, we conclude that attackers are concentrating on targeting the protocol with which they are most familiar. Data analysis also showed that MODBUS traffic was concentrated only on specific dates by a few different source IP addresses using multiple ports to make repeated requests. Also, a significant number of malformed packets were present. From these observations, we assume that the main motive of our attackers was to perform footprinting to collect information rather than exploiting the vulnerabilities of industrial-control-system protocols. Examination of Wireshark captures of EtherNet/IP and MODBUS activities showed that Conpot was not very successful in distinguishing between real protocol activities and embedded protocol requests sent to the ports on which Conpot listened. Since Conpot only logs basic protocol flow data, embedded payloads go unnoticed unless a full packet inspection is performed. Since Conpot can emulate various types of industrial control systems, we conclude that Conpot may provide a quick way to extract signatures for indicators of compromise even if they may not be perfect.

B. FUTURE WORK

To further confirm the practicality of Conpot, different templates can be launched to collect more data. A good way to generate more traffic to Conpot includes posting on websites related to the industrial control systems to advertise IP addresses hosting Conpot. Running several Conpot installations simultaneously may also be effective in leading the attackers to believe that they have discovered a large network of industrial control systems with multiple master servers.

To define a better method to extract signatures of malicious activities on industrial control systems, we suggest using a high-interaction honeypot. A high-interaction honeypot such as GridPot could provide more detailed information about cyberattacks beyond what Conpot provides. GridPot provides more steps of interactions, which will encourage the attackers to believe they have discovered a real industrial control system rather than a honeypot. Complex interactions will allow the researcher to gather more behavioral information from the attackers. From this, the researchers may be able to extract more precise signatures.

APPENDIX. CONPOT INSTALLATION

This appendix summarizes the steps required to install Conpot version 0.5.1.

1. Installing dependencies

```
$ sudo apt-get install libsmi2ldbl snmp-mibs-downloader python-dev libevent-dev  libxslt1-dev libxml2-dev

$ sudo apt-get install libmysqlclient-dev

$ sudo apt-get install python-pip

$ sudo pip install --upgrade pip

$ sudo pip install bacpypes==0.13.8

$ sudo pip install -U pysnmp==4.3.9
```

2. Pulling Conpot from GitHub

```
$ sudo git clone https://github.com/mushorg/conpot.git
```

3. Installing Conpot

```
$ cd Conpot

$ sudo python setup.py install
```

4. Running Conpot with the default template

```
$ sudo conpot --template default
```

Following is a sample console output if Conpot is successfully started.

```
 _
 ___ ___ ___ ___ ___| |_
 | _| . | | . | . | _|
 |___|___|_|_| _|___|_|
 |_|

 Version 0.5.1
 MushMush Foundation

2018-03-05 14:10:00,530 Starting Conpot using template: /usr/local/lib/python2.7/dist-packages/conpot/templates/default
2018-03-05 14:10:00,530 Starting Conpot using configuration found in: /usr/local/lib/python2.7/dist-packages/conpot/conpot.cfg
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] E. Knapp and J. Langill, *Industrial Network Security*. 2nd ed. Waltham, MA, USA: Syngress, 2015. [Online]. Safari Books Online.

[2] T. Macaulay and B. Singer, *Cybersecurity for Industrial Control Systems*. Boca Raton, FL, USA: CRC Press, 2012. [Online]. Safari Books Online.

[3] "About," CONPOT ICS/SCADA Honeypot. Accessed December 11, 2017. [Online]. Available: http://conpot.org/

[4] R. Joshi and A. Sardana, "Honeypots" in *Honeypots: A New Paradigm to Information Security*. Boca Raton, FL, USA: CRC Press, 2011, pp. 1–37.

[5] M. Nawrocki, M, Wahlisch, T. Schmidt, C. Keil, and J. Schonfelder, "A Survey on Honeypot Software and Data Analysis," Cornell University Library. 2016. [Online]. Available: https://arxiv.org/pdf/1608.06249

[6] N. Rowe and J. Rrushi, "Software engineering of deceptive software and systems" in *Introduction to Cyberdeception*. Cham, ZG, Switzerland: Springer International Publishing, 2016, pp. 220–244.

[7] GridPot: Symbolic Cyber-Physical Honeynet Framework. Accessed January 15, 2018. [Online]. Available: http://gridpot.org/

[8] C. Hurd and M. McCarty, "A Survey of Security Tools for the Industrial Control System Environment," Idaho National Laboratory, Idaho Falls, Idaho, USA, 2017. [Online]. Available: https://www.osti.gov/scitech/servlets/purl/1376870

[9] I. Mokube and M. Adams, "Honeypots: Concepts, Approaches, and Challenges," in *Proceedings of the 45th Annual Southeast Regional Conference*, 2007, pp. 321–326.

[10] A. Dominguez, "The State of Honeypots: Understanding the Use of Honey Technologies Today," SANS Institute InfoSec Reading Room, 2017. [Online]. Available: https://www.sans.org/reading-room/whitepapers/detection/state-honeypots-understanding-honey-technologies-today-38165

[11] "Introducing Conpot," The Honeynet Project. Accessed December 12, 2017. [Online]. Available: http://www.honeynet.org/node/1047

[12] "Conpot," ICS/SCADA honeypot. Accessed October 04, 2017. [Online]. Available: https://github.com/mushorg/conpot

[13] K. Wilhoit, “Is Anonymous Attacking Internet Exposed Gas Pump Monitoring Systems in the US?,” TrendLabs Security Intelligence Blog. February 10, 2015. [Online]. Available: http://blog.trendmicro.com/trendlabs-security-intelligence/is-anonymous-attacking-Internet-exposed-gas-pump-monitoring-systems-in-the-us/

[14] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*. 6th ed. Upper Saddle River, NJ, USA: Pearson, 2013.

[15] MODBUS Application Protocol Specification V1.1b3, Modbus, Hopkinton, MA, USA. 2012. [Online]. Available: http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[16] MODBUS Messaging on TCP/IP Implementation Guide V1.0b, Modbus, Hopkinton, MA, USA. 2006. [Online]. Available: http://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

[17] G. Miru, “The Siemens S7 Communication—Part 1 General Structure,” GyM’s Computer Security Rag. January 30, 2016. [Online]. Available: http://gmiru.com/article/s7comm/

[18] G. Miru, “The Siemens S7 Communication—Part 2 Job Requests and Ack Data,” GyM’s Computer Security Rag. June 10, 2017. [Online]. Available: http://gmiru.com/article/s7comm-part2/

[19] L. Brotherson and A. Berlin, “Network Infrastructure” in *Defensive Security Handbook*. 1st ed. Sebastopol, CA, USA: O’Reilly Media, 2017. [Online]. Safari Books Online.

[20] D. Mauro and K, Schmidt, *Essential SNMP*. 2nd ed. Sebastopol, CA, USA: O’Reilly Media, 2005. [Online]. Safari Books Online.

[21] “BACnet®, the ASHRAE building automation and control networking protocol,” ASHRAE. Accessed November 1, 2017. [Online]. Available: https://www.ashrae.org/resources--publications/bookstore/bacnet

[22] “IPMI Basics,” Thomas-Krenn-Wiki. Accessed October 31, 2017. [Online]. Available: https://www.thomas-krenn.com/en/wiki/IPMI_Basics

[23] C. Bodungen, B. Singer, A. Shbeeb, K. Wilhoit and S. Hilt. *Hacking Exposed Industrial Control Systems: ICS and SCADA Security Secrets and Solutions*. New York, NY, USA. 2016. [Online]. Safari Books Online.

[24] *The CIP Networks Library Volume 1: Common Industrial Protocol*, 3rd ed., Open DeviceNet Vendor Association, Inc., Ann Arbor, MI, USA. April 2017.

[25] *CIP Networks Library Volume 2: EtherNet/IP Adaptation of CIP*, 1st ed., Open DeviceNet Vendor Association, Inc., Ann Arbor, MI, USA. April 2017.

[26] C. Sanders and J. Smith. “Detection Mechanisms, Indicators of Compromise, and Signatures” in *Applied Network Security Monitoring*. Waltham, MA, USA: Syngress, 2014. [Online]. Safari Books Online.

[27] Industrial Control Systems Cyber Emergency Response Team, “ICS-CERT Monitor,” Washington, DC, USA, Rep. ICS-MM201306, 2013. [Online]. Available: https://ics-cert.us-cert.gov/monitors/ICS-MM201306

[28] O. Yuksel, J. Hartog, and S. Etalle, “Reading Between the Fields: Practical, Effective Intrusion Detection for Industrial Control Systems,” in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016. [Online]. doi: 10.1145/2851613.2851799

[29] O. Redwood, J. Lawrence, and M. Burmester, “A symbolic honeynet framework for SCADA system threat intelligence,” in *IFIP International Federation for Information Processing*, 2015. [Online]. doi: 10.1007/978-3-319-26567-4-7

[30] R. Piggin, and I. Buffey, “Active defence using an operational technology honeypots,” ATKINS Global, Euston Tower, London, UK, 2016. [Online]. Available: http://www.atkinsglobal.com/~/media/Files/A/Atkins-Corporate/uk-and-europe/services-documents/cyber/Active%20defence%20with%20an%20OT%20honeypot.pdf

[31] A. Serbanescu, S. Obermeier, and D. Yu, “ICS Threat Analysis Using a Large-Scale Honeynet,” in *Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research*, 2015. [Online]. doi: 10.14236/ewic/ICS2015.3

[32] S. Kuman, S. Groš and M. Mikuc, “An experiment in using IMUNES and Conpot to emulate honeypot control networks,” in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, 2017, pp. 1262–1268.

[33] “About,” Wireshark. Accessed March 1, 2018. [Online]. Available: https://www.wireshark.org/

[34] “PDU type,” BACnet Wiki. Accessed March 6, 2018. [Online]. Available: http://www.bacnetwiki.com/wiki/index.php?title=PDU_Type

[35] “Siemens SICAM,” Advisory (ICSA-17-320-02), 2017. [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-17-320-02

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California